You Only Watch Once: A Unified CNN Architecture for Real-Time Spatiotemporal Action Localization

Okan Köpüklü*, Xiangyu Wei*, Gerhard Rigoll

Technical University of Munich, Germany

Abstract

Spatiotemporal action localization requires incorporation of two sources of information into the designed architecture: (1) Temporal information from the previous frames and (2) spatial information from the key frame. Current state-of-the-art approaches usually extract these information with separate networks and use an extra mechanism for fusion to get detections. In this work, we present YOWO, a unified CNN architecture for real-time spatiotemporal action localization in video stream. YOWO makes use of a single neural network to extract temporal and spatial information concurrently and predict bounding boxes and action probabilities directly from video clips in one evaluation. Since the whole architecture is unified, it can be optimized end-to-end. The YOWO architecture is fast providing 34 frames-per-second on 16-frames input clips and 62 frames-per-second on 8-frames input clips. Remarkably, YOWO outperforms the previous state-of-the-art results on J-HMDB-21 (71.1%) and UCF101-24 (75.0%) with 74.4% and 87.2% frame-mAP, respectively.

1. Introduction

The topic of spatiotemporal human action localization has been spotlighted in recent years, which aims to not only recognize the occurrence of an action but also localize it in both time and space. In such a task, comparing with object detection in static images, temporal information plays an essential role. Finding an efficient strategy to aggregate spatial as well as temporal features makes the problem even more challenging. On the other hand, real-time human action detection is becoming increasingly crucial in numerous vision applications, such as human-computer interaction (HCI) systems, unmanned aerial vehicle (UAV) monitoring, autonomous driving, and urban security systems. Therefore, it is desirable and worthwhile to explore a more efficient framework to tackle this problem.

Inspired by the remarkable object detection architecture Faster R-CNN [27], most state-of-the-art works [13] [24] extend the classic two-stage network architecture to action detection, where a number of proposals are produced in the first stage, then classification and localization refinement are performed in the second stage. However, these two-stage pipelines have three main shortcomings in the spatiotemporal action localization task. Firstly, the generation of action tubes which consist of bounding boxes across frames is much more complicated and time-consuming than 2D case. The classification performance is extremely dependent on these proposals, where the detected bounding boxes might be sub-optimal for the following classification task. Secondly, the action proposals focus only on features of humans in the video, neglecting the relationship between humans and some attributes in the background, which yet is able to provide considerably crucial context information for action prediction. The third problem of a two-stage architecture is that training the region proposal network and the classification network separately does not guarantee to find the global optimum. Instead, only local optimum from the combination of two stages can be found. The training cost is also higher than single-stage networks, hence it takes longer time and needs more memory.

1 https://github.com/wei-tim/YOWO
*equal contribution
In this paper, we propose a novel single-stage framework, YOWO (You Only Watch Once), for spatiotemporal action localization in videos. YOWO prevents all of the three shortcomings mentioned above with a single-stage architecture. The intuitive idea of YOWO arises from human’s visual cognitive system. For example, when we are absorbed into the story of a soap opera in front of the TV, each time our eyes capture a single frame. In order to understand which action each artist is performing, we have to relate current frame information (2D features from key frame) to the obtained knowledge from previous frames saved in our memory (3D features from clip). Afterwards, these two kinds of features are fused together to provide us with a reasonable conclusion. The example in Fig. 1 illustrates our inspiration.

YOWO architecture is a single-stage network with two branches. One branch extracts the spatial features of the key frame (i.e. current frame) while the other branch models the spatiotemporal features of the clip consisting of previous frames. In order to aggregate these features smoothly, the channel fusion and attention mechanism is introduced, where we get the utmost out of inter-channel dependencies. Finally, we produce frame-level detections using the fused features, and provide a linking algorithm to generate action tubes.

We carry out comprehensive experiments on J-HMDB-21 and UCF101-24 benchmarks and outperform state-of-the-art results with an improvement of 3.3% and 12.2% improvement on frame-mAP, respectively. We achieve these results operating only on RGB modality and maintaining the real-time capability, which contains utmost importance for real-world applications.

Contributions of this paper are summarized as follows:

(i) We propose a real-time single-stage framework for spatio-temporal action localization in video streams, named YOWO, which can be trained end-to-end with high efficiency. To the best of our knowledge, this is the first work which achieves bounding box regression on features extracted by a 2D-CNN and 3D-CNN, concurrently. These two kinds of features have a complementary effect to each other for the final bounding box regression and action classification.

(ii) We propose a channel fusion and attention mechanism (CFAM) to aggregate the features smoothly from two branches above. CFAM models the inter-channel relationship within the concatenated feature maps and boosts the performance significantly by fusing features more reasonably.

(iii) We perform a detailed ablation study on the YOWO architecture. We examined the effect of 3D-CNN, 2D-CNN, their aggregation and the fusion mechanism. Moreover, we have experimented different 3D-CNN architectures and different clip lengths to explore a further trade-off between the precision and speed.

(iv) We evaluate YOWO on the UCF101-24 and J-HMDB-21 datasets. We experimentally observe that the proposed architecture outperforms state-of-the-art frame-mAP results on both datasets significantly. We also get very competitive results on video-mAP compared to the state-of-the-art results.

2. Related Work

Action recognition with deep learning. Since deep learning brings significant improvements in image recognition, numerous recent research efforts have been devoted to extend it for action recognition in videos. For action recognition, however, besides spatial features extracted from each individual image, temporal context across these frames also needs to be taken into account. Two-stream CNN is one effective strategy to extract spatial and temporal features separately and aggregate them together [6] [30] [36]. Most of these works are based on optical flow, which requires significant computational power to extract, resulting in a time-consuming process. An alternative option to integrate CNN features over time is the implementation of recurrent networks, whose performance, however, is not so satisfying as recent CNN-based methods [42]. 3D-CNNs have been increasingly explored in video analysis tasks recently, which learns the features from both spatial and temporal dimensions simultaneously. 3D-CNN is first exploited to extract spatiotemporal features in [16] and some effective network architectures like C3D [34] and I3D [2] are explored. Inspired by the 2D-CNN residual networks [40], skip connections over layers are also applied to 3D-CNNs to overcome the problem of vanishing gradients [12]. However, 3D-CNN architectures have much more parameters compared to 2D-CNNs, making them computationally expensive. In [18], 3D versions of some famous resource efficient CNN architectures are investigated. For resource efficiency, some other works focus on learning 2D features from single images with a 2D-CNN and then fusing them together to learn temporal features with a 3D-CNN [43].

Spatiotemporal action localization. For object detection in images, R-CNN series extract region proposals using selective search [10] or RPN [27] in the first stage and classify the objects in these potential regions in the second stage. Despite faster R-CNN [27] achieves state-of-the-art results in object detection, it is hard to implement it for real-time tasks due to its time-consuming two-stage architecture. Meanwhile, YOLO [25] and SSD [23] aim to simplify this process to one stage and have outstanding real-time performance. For action localization in videos, due to the success of R-CNN series most of the research approaches propose first detecting the humans in each frame and then linking these bounding boxes reasonably as action tubes [11, 24, 13]. Two-stream detectors introduce an additional
stream on the base of the original classifier for optical flow modality [24] [29] [32]. Some other works produce clip tube proposals with 3D-CNNs and achieve regression as well as classification on the corresponding 3D features [13] [29], thus region proposal is necessary for them. In a recent work [4], authors propose a 3D capsule network for video action detection which can jointly perform pixel-wise action segmentation along with action classification. However, it is too expensive in terms of computational complexity and number of parameters since it is a U-Net [28] based 3D-CNN architecture.

**Attention modules.** Attention is an effective mechanism to capture long-range dependencies and has been attempted to be used in CNNs to boost the performance in image classification [35] [3] [39] and scene segmentation [7]. Attention mechanism is implemented spatial-wise and channel-wise in these works, in which spatial attention addresses the inter-spatial relationship among features while channel attention enhances the most meaningful channels and weakens the others. As a remarkable work of channel-wise attention, Squeeze-and-Excitation module [14] is beneficial to increase CNN’s performance with little computational cost. On the other hand, for video classification tasks, non-local block [37] takes spatio-temporal information into account simultaneously to learn the dependencies of features across frames, which can be viewed as a self-attention strategy.

Different from previous works, we extend YOLO [25] in the task of spatio-temporal action localization and design a two-stream model to analyze the spatial and temporal features simultaneously. We name it as YOWO as we make use of a clip only once and detect the corresponding actions in the key frame. However, to avoid the complex optical flow computation, we use 2D features of the key frame and 3D features of the clip together. Afterwards, these two kinds of features are fused together carefully with the application of attention mechanism such that rich contextual relationships are well taken into account.

### 3. Methodology

In this section, we first present YOWO’s architecture in detail, which extracts 2D features from the key frame as well as 3D features from the input clip concurrently and aggregates them together. Then the implementation of channel fusion and attention mechanism is discussed, which provides the essential performance boost. Finally we describe the details of the training process for the YOWO architecture and the improved bounding box linking strategy for generation of action tubes in untrimmed videos.

#### 3.1. YOWO architecture

The YOWO architecture is illustrated in Fig. 2, which can be divided into four major parts: 3D-CNN, 2D-CNN, CFAM and bounding box regression parts.

**3.1.1 3D-CNN**

Since contextual information is crucial for human action understanding, we utilize 3D-CNN for action recognition. 3D-CNNs are able to capture motion information by applying convolution operation not only in space dimension but also in time dimension. The basic 3D-CNN architecture in our framework is 3D-ResNext-101 due to its high performance in Kinetics dataset [12]. In addition to 3D-ResNext-101, we have also experimented with different 3D-CNN models in our ablation study. For all 3D-CNN architectures, all of the layers after the last conv layer are discarded. The input to the 3D network is a clip of a video, which is composed of a sequence of successive frames in time order, and has a shape of $[C \times D \times H \times W]$, while the last conv layer of 3D ResNext-101 outputs a feature map of shape $[C' \times D' \times H' \times W']$ where $C' = 3$, $D'$
is the number of input frames, \( H \) and \( W \) are height and width of input images, \( C' \) is the number of output channels, \( D' = 1 \), \( H' = \frac{H}{32} \) and \( W' = \frac{W}{32} \). The depth dimension of the output feature map is reduced to 1 such that output volume is squeezed to \([C' \times H' \times W']\) in order to match the output feature map of 2D-CNN.

### 3.1.2 2D-CNN

In the meantime, to address the spatial localization problem, 2D features of the key frame are also extracted in parallel. We employ Darknet-19 [26] as the basic backbone of our 2D network due to its good balance between accuracy and efficiency. The key frame with the shape \([C \times H \times W]\) is the most recent frame of the input clip, thus there is no need for an additional data loader. The output feature map of Darknet-19 has a shape of \([C' \times H' \times W']\) where \( C = 3 \), \( C'' \) is the number of output channels, \( H' = \frac{H}{32} \) and \( W' = \frac{W}{32} \). The output feature map \( \text{2D-CNN} \) is the total number of channels in concatenated features, is firstly fed into two convolutional layers to generate a new feature map \( B \in \mathbb{R}^{C' \times H' \times W'} \). Afterwards, several operations are performed on the feature map \( B \).

Assume \( F \in \mathbb{R}^{C \times N} \) is the reshaped tensor from feature map \( B \), where \( N = H \times W \), which means that features in every single channel is vectorized to one dimension:

\[
B \in \mathbb{R}^{C' \times H' \times W'} \xrightarrow{\text{vectorization}} F \in \mathbb{R}^{C \times N}
\] (1)

Then a matrix product between \( F \in \mathbb{R}^{C \times N} \) and its transpose \( F^T \in \mathbb{R}^{N \times C} \) is performed to produce Gram matrix \( G \in \mathbb{R}^{C \times C} \), which indicates the feature correlations across channels [8]:

\[
G = F \cdot F^T \quad \text{with} \quad G_{ij} = \sum_{k=1}^{N} F_{ik} \cdot F_{jk}
\] (2)

where each element \( G_{ij} \) in the Gram matrix \( G \) represents the inner product between the vectorised feature map \( i \) and \( j \). After computing the Gram matrix, a softmax layer is applied to generate channel attention map \( M \in \mathbb{R}^{C \times C} \):

\[
M_{ij} = \frac{\exp(G_{ij})}{\sum_{j=1}^{C} \exp(G_{ij})}
\] (3)

where \( M_{ij} \) is a score measuring the \( j^{th} \) channel’s impact on the \( i^{th} \) channel. Thus the attention map \( M \) can be viewed as it summaries the inter-channel dependency of features given a feature map. To perform the impact of attention map to original features, a further matrix multiplication between the transpose of \( M \) and \( F \) is carried out and the result is reshaped back to 3-dimensional space \( \mathbb{R}^{C \times H \times W} \), which has the same shape as the input tensor:

\[
F' = M^T \cdot F
\] (4)

\[
F' \in \mathbb{R}^{C \times N} \xrightarrow{\text{reshape}} F'' \in \mathbb{R}^{C \times H \times W}
\] (5)

The output of channel attention module \( C \in \mathbb{R}^{C \times H \times W} \) combines this result with the original input feature map \( B \) with a trainable scalar parameter \( \alpha \) using an element-wise sum operation, and \( \alpha \) gradually learns a weight from 0:

\[
C = \alpha \cdot F'' + B
\] (6)

Figure 3: Channel fusion and attention mechanism for aggregating output feature maps coming from 2D-CNN and 3D-CNN branches.
The Eq. (6) shows that the final feature of each channel is a weighted sum of the features of all channels and original features, which models the long-range semantic dependencies between feature maps. Finally, the feature map $C \in \mathbb{R}^{C \times H' \times W'}$ is fed into two more convolutional layers to generate the output feature map $D \in \mathbb{R}^{C' \times H' \times W'}$ of the CFAM module. Two convolutional layers at the beginning and the end of CFAM modules helps to mix the features coming from different backbones and having possibly different distributions.

Such an architecture promotes the feature representativeness in terms of inter-dependencies among channels and thus the features from different dimensions can be aggregated reasonably and smoothly. Besides, Gram matrix takes the whole feature map into consideration, where the dot product of each two flattened feature vectors presents the information about the relation between them. A larger product indicates that the features in these two channels are more correlated while a smaller product suggests that they are different from each other. For a given channel, we allocate more weights to the other channels which are much correlated and have more impact to it. By means of this mechanism, contextual relationship is emphasized and feature discriminability is enhanced.

### 3.1.4 Bounding box regression

We follow the same guidelines of YOLO [26] for bounding box regression. A final convolutional layer with $1 \times 1$ kernels is applied to generate desired number of output channels. For each grid cell in $H' \times W'$, 5 prior anchors are selected by k-means technique on corresponding datasets with $NumCls$ class conditional action scores, 4 coordinates and confidence score making the final output size of YOWO $[(5 \times (NumCls + 5)) \times H' \times W']$. The regression of bounding boxes are then refined based on these anchors.

We have used multi-scale training while the resolution of each frame is set to 224 x 224 at test time. We select the mini-batch stochastic gradient decent algorithm with momentum and weight decay strategy to optimize the loss function, which is defined similar to the original YOLO network [26] except that we apply smooth $L_1$ loss for localization as in [9] since it is less sensitive to outliers than the $L_2$ loss and focal loss [21] for classification loss.

### 3.2. Implementation details

We initialize the 3D and 2D network parameters separately: 3D part with pretrained models on Kinetics [2] and 2D part with pretrained models on PASCAL VOC [22]. Although our architecture consists of 2D-CNN and 3D-CNN branches, the parameters are able to be updated jointly. The learning rate is initialized as 0.0001 and reduced with a factor of 0.5 after 30k, 40k, 50k and 60k iterations. For the dataset UCF101-24, the training process is completed after 5 epochs while for J-HMDB-21 after 10 epochs. The complete architecture is implemented and trained end-to-end in PyTorch.

In the trainings, because of the small number of samples in J-HMDB-21, we freeze all the 3D conv network parameters thus the convergence is faster and over-fitting risk can be reduced. In addition, for both UCF101-24 as well as J-HMDB-21, we deploy several data augmentation techniques such as flipping images horizontally in the clip, random scaling and random spatial cropping. During testing, only detected bounding boxes with confidence score larger than threshold 0.25 are selected and then post-processed with non-maximum suppression with a threshold of 0.4.

### 3.3. Linking

As we have already obtained frame-level action detections, next step is to link these detected bounding boxes to construct action tubes in the whole video. We apply the similar linking algorithm as described in [11] [24] to simply find the optimal video-level action detections.

Assume $R_t$ and $R_{t+1}$ are two regions from consecutive frames $t$ and $t+1$, the linking score for an action class $c$ is defined as

$$s_c(R_t, R_{t+1}) = \psi(ov) \cdot [s_c(R_t) + s_c(R_{t+1})]$$

$$+ \alpha \cdot s_c(R_t) \cdot s_c(R_{t+1})$$

$$+ \beta \cdot ov(R_t, R_{t+1})$$

(7)

where $s_c(R_t), s_c(R_{t+1})$ are class specific scores of regions $R_t$ and $R_{t+1}, ov$ is the intersection-over-union of these two regions, $\alpha$ and $\beta$ are scalars. $\psi(ov)$ is a constraint which is equal to 1 if an overlap exists ($ov > 0$), otherwise $\psi(ov)$ is equal to 0. We extend the linking score definition in [24] with an extra element $\alpha \cdot s_c(R_t) \cdot s_c(R_{t+1})$, which takes the dramatic change of scores between two successive frames into account and is able to improve the performance of video detection in experiments. After all the linking scores are computed, Viterbi algorithm is deployed to find the optimal path to generate action tubes. More details are described in [24].

### 4. Experiments

To evaluate YOWO’s performance, two popular and challenging action detection datasets, UCF101-24 [33] and J-HMDB-21 [15] are selected. We follow the official evaluation metrics strictly to report the results and compare the performance of our method with the state of the art. Moreover, we also do a detailed ablation study in order to explore characteristics of YOWO architecture and contribution of each building block to its performance.
4.1. Datasets and evaluation metrics

**UCF101-24** is a subset of UCF101 [33], which is originally an action recognition dataset of realistic action videos. UCF101-24 contains 24 action classes and 3207 videos, for which the corresponding spatio-temporal annotations are provided. In addition, there might be multiple action instances in each video, which have the same class label but different spatial and temporal boundaries. Such a property makes video-level action detection much more challenging. As in previous works, we perform all the experiments on the first split.

**J-HMDB-21** is a subset of the HMDB-51 dataset [20] and consists of 928 short videos with 21 action categories in daily life. Each video is well trimmed and has a single action instance across all the frames. We report our experimental results on the first split.

**Evaluation metrics:** We employ two popular metrics used by the most researches in the region of spatio-temporal action detection to generate convincing evaluations. Following strictly the rule applied by the PASCAL VOC 2012 metric [5], frame-mAP measures the area under the precision-recall curve of the detections for each frame. On the other hand, video-mAP focuses on the action tubes [11]. If the mean per frame intersection-over-union with the ground truth across the frames of the whole video is greater than a threshold and in the meanwhile the action label is correctly predicted, then this detected tube is regarded as a correct instance. Finally, the average precision for each class is computed and the average over all classes is reported.

<table>
<thead>
<tr>
<th>Model</th>
<th>UCF101-24</th>
<th>J-HMDB-21</th>
</tr>
</thead>
<tbody>
<tr>
<td>2D</td>
<td>61.7</td>
<td>36.0</td>
</tr>
<tr>
<td>3D</td>
<td>71.5</td>
<td>41.5</td>
</tr>
<tr>
<td>2D + 3D</td>
<td>77.9</td>
<td>47.1</td>
</tr>
<tr>
<td>2D + 3D + CFAM</td>
<td><strong>85.8</strong></td>
<td><strong>64.9</strong></td>
</tr>
</tbody>
</table>

Table 1: Frame-mAP @ IoU 0.5 results on datasets UCF101-24 and J-HMDB-21 for different models. For all architectures, the input to 3D-CNNs is 8 frames clips with downsampling 1.

<table>
<thead>
<tr>
<th>Model</th>
<th>Localization (recall)</th>
<th>Classif.</th>
</tr>
</thead>
<tbody>
<tr>
<td>UCF101-24</td>
<td>2D</td>
<td>91.7</td>
</tr>
<tr>
<td></td>
<td>3D</td>
<td>90.8</td>
</tr>
<tr>
<td></td>
<td>2D + 3D</td>
<td>93.2</td>
</tr>
<tr>
<td></td>
<td>2D + 3D + CFAM</td>
<td><strong>93.5</strong></td>
</tr>
</tbody>
</table>

| J-HMDB-21      | 2D                    | 94.3     | 50.6     |
|                | 3D                    | 76.3     | 69.3     |
|                | 2D + 3D               | 94.5     | 63.0     |
|                | 2D + 3D + CFAM        | **97.3** | **76.1** |

Table 2: Localization @ IoU 0.5 (recall) and classification results on UCF101-24 and J-HMDB-21. For all architectures, the input to 3D-CNNs is 8 frames clips with downsampling 1.

Moreover, in order to explore the impact of each 2D-CNN, 3D-CNN and CFAM blocks, we investigate the localization and the classification performance of different architectures, which is given in Table 2. For localization, we look at the recall value, which is the ratio of the number of correctly localized actions to the total number of proposed detections. For classification, we look at the classification accuracy of the correctly localized detections. For both datasets, 2D network is better at localization while 3D network performs better at classification. It is also obvious that CFAM module boosts both localization and classification performance.

**How many frames are suitable for temporal information?** For 3D-CNN branch, different clip lengths with different downsampling rates can change the performance of overall YOWO architecture [19]. Therefore, we conduct experiments with 8-frames and 16-frames clips with different downsampling rates, which is given in Table 3. For example, 8-frames (d=3) refers to selecting 8 frames from 24 frames window with downsampling rate of 3. Specifically, we compare three downsampling rates $d = 1, 2, 3$ for clip length 8-frames and two downsampling rates $d = 1, 2$ for 16-frames clip length.

As expected, we observe that the framework with input of 16 frames performs better than 8 frames since long frame sequence contains more temporal information. However, as
Table 3: Frame-mAP @ IoU 0.5 results on datasets UCF101-24 and J-HMDB-21 for different clip lengths and different downsampling rates $d$.

<table>
<thead>
<tr>
<th>Input</th>
<th>UCF101-24</th>
<th>J-HMDB-21</th>
</tr>
</thead>
<tbody>
<tr>
<td>8-frames (d=1)</td>
<td>85.8</td>
<td>64.9</td>
</tr>
<tr>
<td>8-frames (d=2)</td>
<td>84.4</td>
<td>61.5</td>
</tr>
<tr>
<td>8-frames (d=3)</td>
<td>84.3</td>
<td>61.0</td>
</tr>
<tr>
<td>16-frames (d=1)</td>
<td><strong>87.2</strong></td>
<td><strong>74.4</strong></td>
</tr>
<tr>
<td>16-frames (d=2)</td>
<td>85.1</td>
<td>71.4</td>
</tr>
</tbody>
</table>

Table 4: Frame-mAP @ IoU 0.5 results on datasets UCF101-24 and J-HMDB-21 for different 3D backbones. For all architectures, the input to 3D-CNNs is 16 frames (d=1) clips.

<table>
<thead>
<tr>
<th>Model</th>
<th>UCF101-24</th>
<th>J-HMDB-21</th>
</tr>
</thead>
<tbody>
<tr>
<td>3D-ResNext-101</td>
<td><strong>87.2</strong></td>
<td><strong>74.4</strong></td>
</tr>
<tr>
<td>3D-ResNet-101</td>
<td>86.0</td>
<td>70.8</td>
</tr>
<tr>
<td>3D-ResNet-50</td>
<td>85.9</td>
<td>61.3</td>
</tr>
<tr>
<td>3D-ResNet-18</td>
<td>72.6</td>
<td>39.3</td>
</tr>
<tr>
<td>3D-ShuffleNetV1 2.0x</td>
<td>68.8</td>
<td>37.5</td>
</tr>
<tr>
<td>3D-ShuffleNetV2 2.0x</td>
<td>63.3</td>
<td>36.7</td>
</tr>
<tr>
<td>3D-MobileNetV1 2.0x</td>
<td>68.6</td>
<td>36.7</td>
</tr>
<tr>
<td>3D-MobileNetV2 1.0x</td>
<td>68.5</td>
<td>39.4</td>
</tr>
</tbody>
</table>

Table 5: Performance on dataset J-HMDB-21 and comparison with SOTA results by frame-mAP (%) under IOU threshold 0.5 and video-mAP (%) under different IOU thresholds.

<table>
<thead>
<tr>
<th>Method</th>
<th>Frame-mAP</th>
<th>Video-mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peng w/o MR [24]</td>
<td>65.9</td>
<td>51.1</td>
</tr>
<tr>
<td>Peng w/ MR [24]</td>
<td>58.5</td>
<td>42.3</td>
</tr>
<tr>
<td>ROAD [32]</td>
<td>-</td>
<td>73.5</td>
</tr>
<tr>
<td>T-CNN [13]</td>
<td>61.3</td>
<td>47.1</td>
</tr>
<tr>
<td>ACT [17]</td>
<td>63.3</td>
<td>47.1</td>
</tr>
<tr>
<td>P3D-CTN [38]</td>
<td>71.1</td>
<td>51.4</td>
</tr>
<tr>
<td>TPNet [31]</td>
<td>-</td>
<td>72.4</td>
</tr>
<tr>
<td>YOWO (16-frame)</td>
<td><strong>74.4</strong></td>
<td><strong>87.8</strong></td>
</tr>
</tbody>
</table>

Table 6: Performance on dataset UCF101-24 and comparison with SOTA results by frame-mAP (%) under IOU threshold 0.5 and video-mAP (%) under different IOU thresholds.

<table>
<thead>
<tr>
<th>Method</th>
<th>Frame-mAP</th>
<th>Video-mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peng w/o MR [24]</td>
<td>56.9</td>
<td>42.2</td>
</tr>
<tr>
<td>Peng w/ MR [24]</td>
<td>58.5</td>
<td>42.3</td>
</tr>
<tr>
<td>ROAD [32]</td>
<td>-</td>
<td>73.5</td>
</tr>
<tr>
<td>T-CNN [13]</td>
<td>61.3</td>
<td>47.1</td>
</tr>
<tr>
<td>ACT [17]</td>
<td>63.3</td>
<td>47.1</td>
</tr>
<tr>
<td>MPS [1]</td>
<td>51.1</td>
<td>47.1</td>
</tr>
<tr>
<td>YOWO (16-frame)</td>
<td><strong>87.2</strong></td>
<td><strong>82.5</strong></td>
</tr>
</tbody>
</table>

4.3. State-of-the-art comparison

We have compared YOWO with other state-of-the-art architectures on J-HMDB-21 and UCF101-24 datasets. For the sake of fairness, we have excluded VideoCapsuleNet [4] as it uses different video-mAP calculation without constructing action tubes via some linking strategies. However, YOWO still performs 9.8% and 8.6% better than VideoCapsuleNet in terms of frame-mAP @ 0.5 IoU on J-HMDB-21 and UCF101-24, respectively.

4.3.1 Performance comparison on J-HMDB-21

YOWO is compared with the previous state-of-the-art methods on J-HMDB-21 in Table 5. Using the standard metrics, we report the frame-mAP at IOU threshold 0.5 and the video-mAP at various IOU thresholds. YOWO consistently outperforms the state-of-the-art results on dataset J-HMDB-21, with a frame-mAP increase of 3.3% and a video-mAP increase of 3.8%, 5.2% at IOU thresholds of 0.2 and 0.5, respectively.

4.3.2 Performance comparison on UCF101-24

Table 6 presents the comparison of YOWO with the state-of-the-art methods on UCF101-24. YOWO achieves 87.2% with respect to frame-mAP metric, which is significantly better than the others by preceding the second best result.
Figure 4: Sample action localizations for UCF101-24 and J-HMDB-21. Red bounding boxes are ground truth while green and orange are true and false positive localizations, respectively.

<table>
<thead>
<tr>
<th>Model</th>
<th>Run time (fps)</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F-mAP</td>
<td>V-mAP</td>
</tr>
<tr>
<td>Saha et al. [29]</td>
<td>4</td>
<td>36.4</td>
</tr>
<tr>
<td>ROAD (A) [32]</td>
<td>40</td>
<td>40.9</td>
</tr>
<tr>
<td>ROAD (A+RTF) [32]</td>
<td>28</td>
<td>41.9</td>
</tr>
<tr>
<td>ROAD (A+AF) [32]</td>
<td>7</td>
<td>46.3</td>
</tr>
<tr>
<td>YOWO-ResNext-101 (8-frames)</td>
<td>62</td>
<td>85.8 47.6</td>
</tr>
<tr>
<td>YOWO-ResNext-101 (16-frames)</td>
<td>34</td>
<td>87.2 48.8</td>
</tr>
</tbody>
</table>

Table 7: Run time and performance on dataset UCF101-24 (16 frames, d=1). The IoU thresholds for frame-mAP and video-mAP are set to 0.5.

However, YOWO also has some drawbacks. Firstly, since YOWO captures all the content of the key frame and the clip, it sometimes makes some false positive detections before the actions are performed. For example, in Fig. 4 first row last image, YOWO sees a person holding a ball at a basketball court and consequently recognizes him very confidently although he is not shooting the ball yet. Secondly, YOWO needs enough temporal content to make correct action localization. If an actor starts performing action suddenly, localization at initial frames lacks temporal content and false actions are recognized consequently, as in Fig. 4 second row last image (climbing stair instead of running).

4.3.3 Runtime comparison

Most of the state-of-the-art methods are two stage architectures, which are computationally expensive to run in real time. YOWO is a unified architecture, which can be trained end-to-end. In addition, we do not employ optical flow, which is computationally burdensome. In Table 7, we compare runtime performance of YOWO with other state-of-the-art methods. YOWO’s speed is calculated in terms of frames per second (fps) on a single NVIDIA Titan Xp GPU with a batch size of 8. It must be noted that YOWO’s 2D and 3D backbones can be replaced with any arbitrary CNN model according to the needs.

4.4. Model visualization

In general, YOWO architecture performs a decent job at localizing actions in videos, which is illustrated in Fig. 4.

4.4. Model visualization

In general, YOWO architecture performs a decent job at localizing actions in videos, which is illustrated in Fig. 4.

However, YOWO also has some drawbacks. Firstly, since YOWO captures all the content of the key frame and the clip, it sometimes makes some false positive detections before the actions are performed. For example, in Fig. 4 first row last image, YOWO sees a person holding a ball at a basketball court and consequently recognizes him very confidently although he is not shooting the ball yet. Secondly, YOWO needs enough temporal content to make correct action localization. If an actor starts performing action suddenly, localization at initial frames lacks temporal content and false actions are recognized consequently, as in Fig. 4 second row last image (climbing stair instead of running).

5. Conclusion

In this paper, we presented a novel unified architecture for spatiotemporal action localization in video streams. Our approach, YOWO, models the spatiotemporal context from successive frames for action understanding while extracting the fine spatial information from key frame to address the localization task in parallel. In addition, we propose a channel fusion and attention mechanism for effective aggregation of these two kinds of information. Since we do not separate human detection and action classification procedures, the whole network can be optimized by a joint loss in an end-to-end framework. We have carried out a series of comparative evaluations on two challenging representative datasets UCF101-24 and J-HMDB-21. Our approach outperforms the other state-of-the-art results while retaining real-time capability, which makes it possible to deploy it on mobile devices.

Acknowledgements

The Titan Xp used for this research was donated by the NVIDIA Corporation.
References


[27] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region


